### Why Linear Algebra for Data Science?

Data Science deals with extracting meaningful insights from the data. We need to represent data and perform numerous operations on them to extract insights. **Linear Algebra** helps in representation and operations on data in Data Science and Machine Learning.

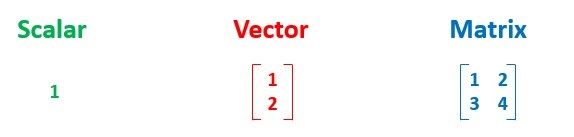
Let’s understand the application of Linear algebra in Data Science with the following examples.

* **Deep Learning**  - Representing input to the model and model parameters as vector and matrices and making calculations using Linear Algebraic operations.
* **Image data** - Representing images as matrices and doing various geometrical transformations on them using Linear Algebraic operations.
* **Recommender system**  - using linear algebraic concepts to measure similarity.
* **Multiple linear regression**   - Solving multiple linear  equations using linear algebra
* **Feature extraction (PCA)** - using the linear algebraic concept of eigenvalue and eigenvectors
* **One hot encoding** - A matrix representation of the encoding

### **What is Linear Algebra?**

Linear Algebra is a branch of mathematics. It provides basic structures to represent data and various numerical methods and tools to solve problems.

The basic building blocks of Linear Algebra are scalar, vector, and matrix. A **Scalar** is a quantity, described by a numeric value. A **Vector** is an ordered collection of scalars. A **Matrix** is a collection of vectors.



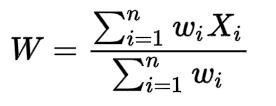
Let us explore an example to understand these building blocks.

Consider a student who has taken 3 exams for a given subject. The scores are 80%, 80%, and 85% respectively. Based on the difficulty level, the professor has allocated weights of 30%, 30%, and 40% to the exams respectively.

### **Problem**

Calculate the weighted mean of the marks obtained by the student.

The weighted mean 'W' is calculated by the formula:



where '**wi**' represent the weights and '**Xi**' represents the corresponding marks.

Here, individual score and weightage are considered as scalars. The collection of the scores and the corresponding weights can be represented as vectors as shown below:

x = [80, 80, 85]

w = [0.3, 0.3, 0.4]

In Linear Algebra, an operation is defined called ‘dot product’ which helps in multiplying two vectors. The dot product of the vectors w and X is:



This dot product will result in finding the weighted mean of the marks (which is a scalar).

Linear Algebraic operations are applied to the entire object (vector/matrix) instead of individual data points one at a time. This technique is called as **Vectorization** and it is more efficient while dealing with operations on large data.

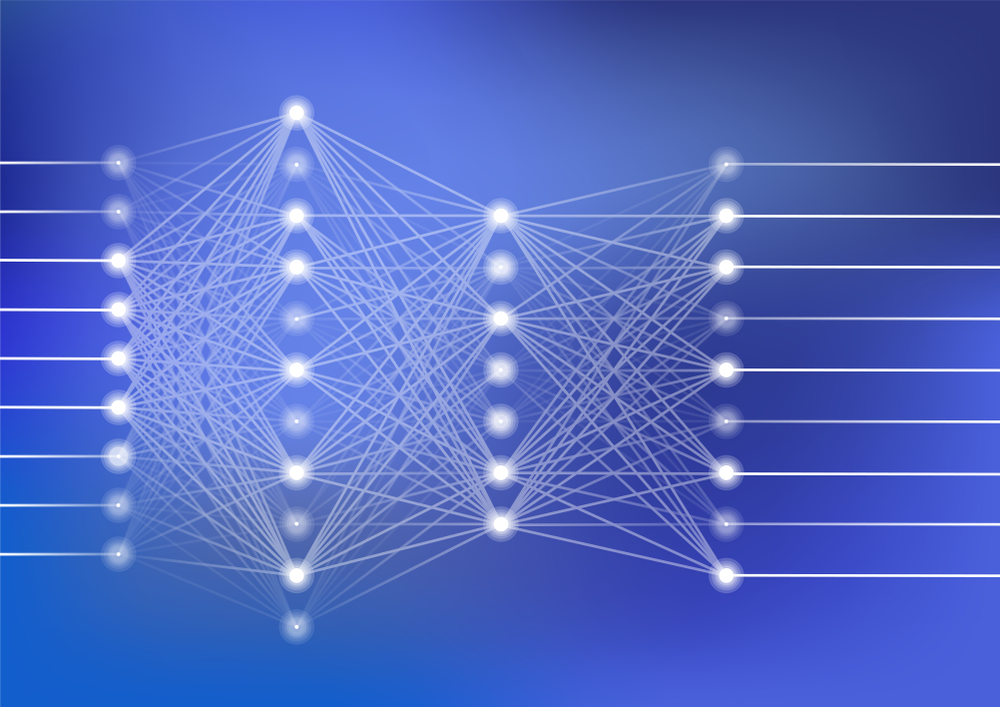
### Deep Learning

Over the past few decades, path-breaking innovations have transformed the way the world functions. Some of the notable innovations are speech recognition, self-driving cars, automated stores etc.

Deep Learning algorithms play a major role in these innovations. They are based on the concept of Artificial Neural Network (ANN) which involves linear algebraic data structures and operations.

Image Processing

An important aspect of Artificial Intelligence is **computer vision**. Most computer vision tasks deal with images represented as an n-dimensional array of pixel values. Linear algebraic tools and techniques help us model and process these images to extract meaningful information.



### Image Processing

An important aspect of Artificial Intelligence is **computer vision**. Most computer vision tasks deal with images represented as an n-dimensional array of pixel values. Linear algebraic tools and techniques help us model and process these images to extract meaningful information.



### **Types of Matrices**

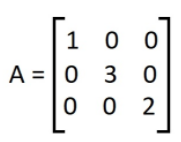
Matrices can be categorized on the basis of the value of their elements, position of elements,  number of rows and columns, etc. as follows:

* Diagonal matrix
* Identity matrix
* Symmetric matrix
* Triangular matrix

### **Diagonal Matrix**

A Square Matrix (D) is called a Diagonal Matrix, if D has zeros outside the main diagonal or principal diagonal. The Main diagonal or the principal diagonal are the elements on the diagonal that runs from the top left to bottom right.

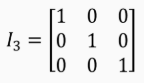
Example: The matrix 'A' shown below illustrates a (3, 3) Diagonal matrix.



### **Identity Matrix**

An identity matrix is a square matrix of dimensions (n, n) having '1' across its main diagonal and '0' everywhere else. It is usually represented as 'In'

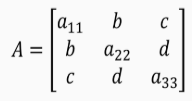
Example: The matrix shown below illustrates a (3, 3) Identity matrix



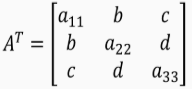
### **Symmetric Matrix**

A square matrix A of dimension (n, n) is symmetric, if A = AT i.e.  the matrix A is the same as its transpose.

Example: Consider the following matrix A:



The Transpose of A (AT) is shown below:

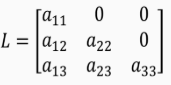


### **Triangular Matrix**

A triangular matrix can be either a lower triangular or an upper triangular matrix.

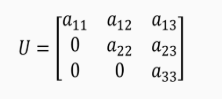
A lower triangular matrix is a square matrix in which all the elements above the main diagonal are zero.

Example: L is a lower triangular matrix of dimension (3, 3)



An upper triangular matrix is a square matrix in which all the elements below the main diagonal are zero.

Example: U is an upper triangular matrix of dimension (3, 3)

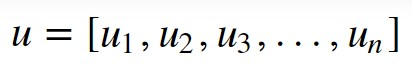


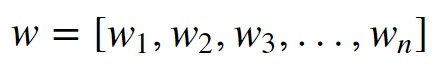
### **OPERATION OVER VECTORS AND MATRICES**

### **Introduction**

Vectors of the same dimension can be added by adding their corresponding elements.

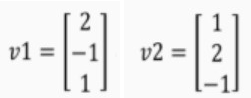
Consider 2 vectors **u** and **w** of dimension n as follows:

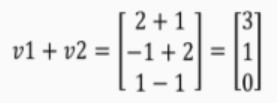






Example: Consider 2 column vectors v1 and v2 as follows:

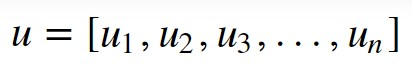


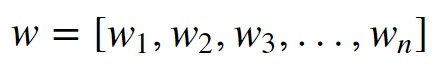


### **Vector Addition**

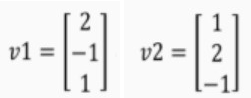
Vectors of the same dimension can be added by adding their corresponding elements.

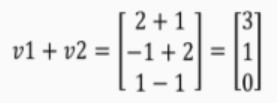
Consider 2 vectors **u** and **w** of dimension n as follows:





Example: Consider 2 column vectors v1 and v2 as follows:

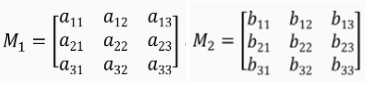


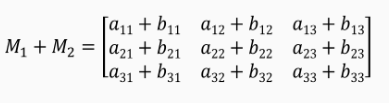


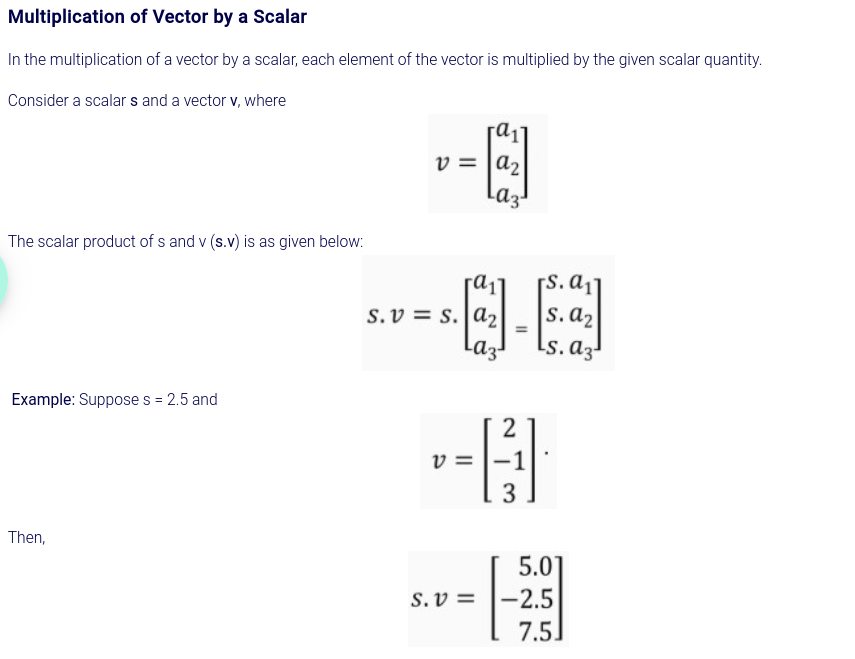
### **Matrix Addition**

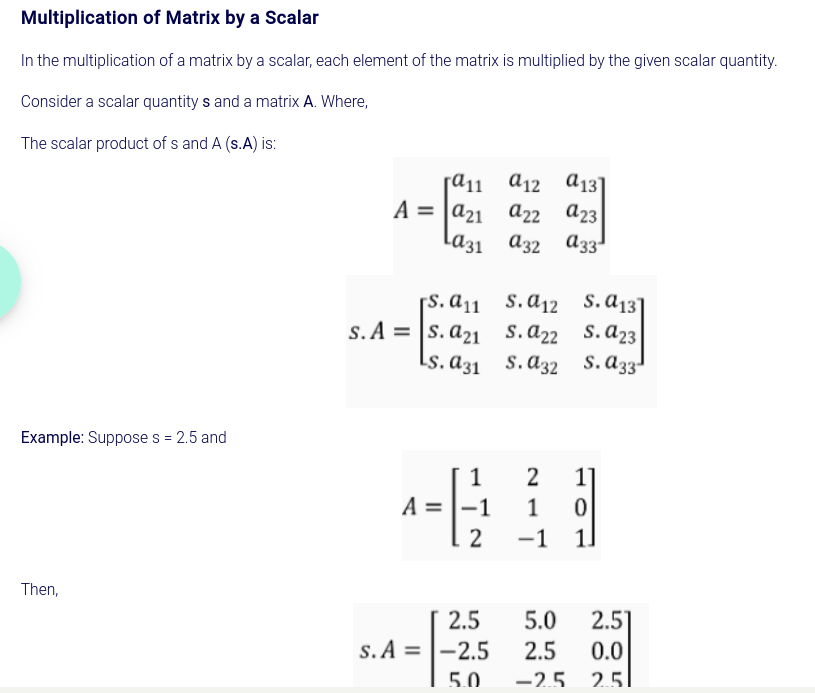
Matrices of the same dimension can be added by adding their corresponding elements.

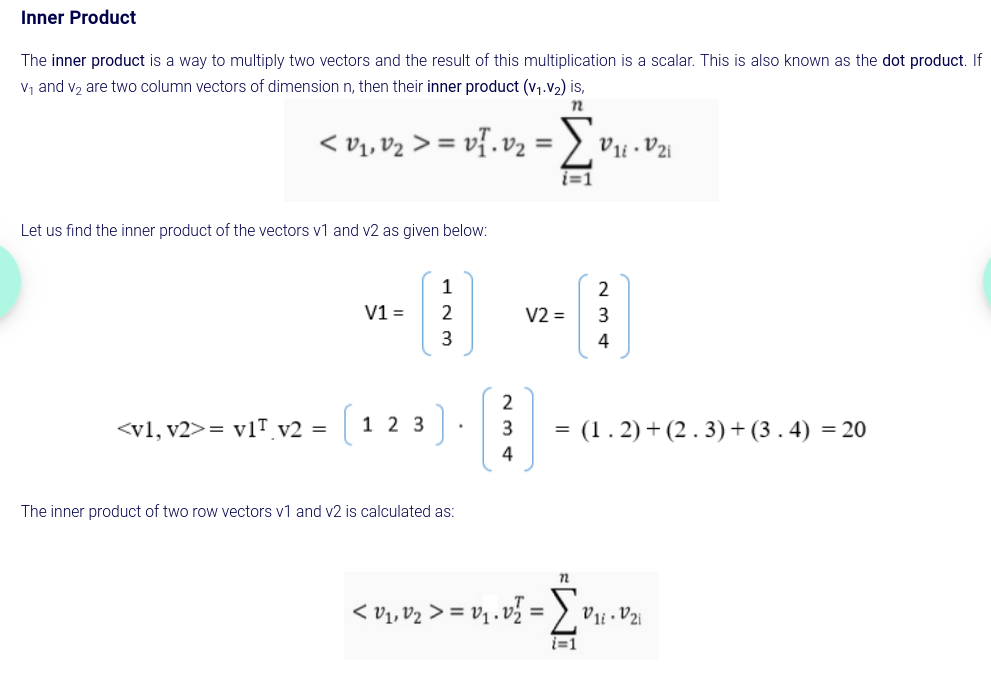
Consider 2 Matrices M1 and M2 of dimension (3, 3) as follows:





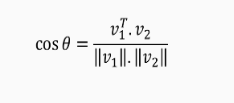


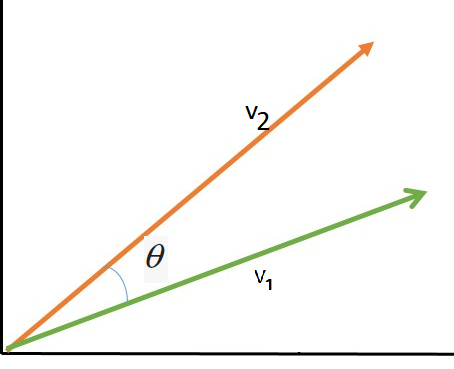




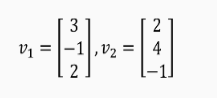
### **Angle between vectors**

If v1 and v2 are non-zero vectors of dimension n then the cosine of the angle between them is:





Example: Define two vectors v1 and v2 as follows:

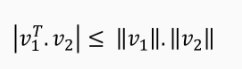


Now, cos **θ** = 0 (as the dot product of v1 and v2 is 0).

Therefore, the angle between the vectors v1 and v2 is 90 degree.

### **Cauchy - Schwarz**

Cauchy-Schwarz inequality is one of the important mathematical inequality defined by the following expression:



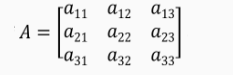
where, v1 and v2 are two vectors of same dimension.

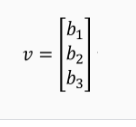
This can be used to check the linear dependence of two vectors.

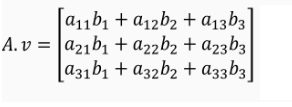
### **Matrix - Vector Multiplication**

Consider a matrix A of dimension (m, n) and a column vector v of dimension n. The product of A and v is represented as A.v, which is a column vector of dimension m.

For example, if A is a matrix of dimension (3, 3) and v is a column vector of dimension 3, then the resultant matrix will be of dimension (3, 1), which is also a column vector.



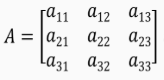


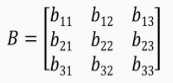


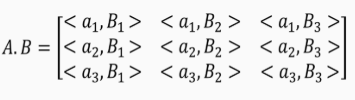
### **Matrix Multiplication**

Consider a matrix A of dimension (m, n) and a matrix B of dimension (p, q). A.B is possible, only if **n = p** and the resultant matrix is of dimension (m, q).

Consider the matrices A and B of dimension (3, 3):

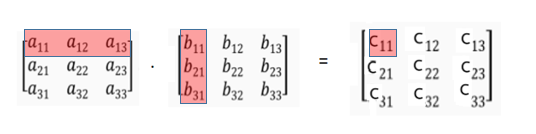






Where, ai is the ith row of the matrix A and Bi is the ith column of Matrix B and <ai, Bi> is the inner product (dot product) of the vectors ai and Bi.

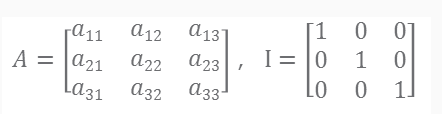
The following image represents <a1, B1> as part of matrix multiplication:



### **Permutation Matrix**

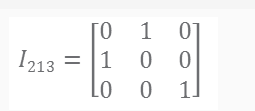
Permutation matrix is obtained by permuting (or interchanging) the rows or columns of an Identity matrix.

Consider a matrix A and Identity matrix I,

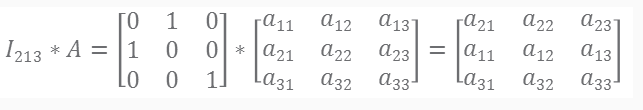


Since I is an Identity matrix,

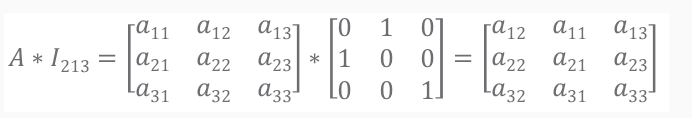
I\*A = A\*I = A.

If I213 is created by swapping Rows 1 and 2 of I then, I213 is:

Pre-multiplying I213 with A results in rows 1 and 2 of A getting swapped:



Post-multiplying I213 with A results in columns 1 and 2 of A getting swapped:

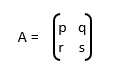


The number of permutation matrices depends on the order of the identity matrix. For an identity matrix of order 3, there are six different permutation matrices (I123, I132, I213, I231, I312, I321 ). For an identity matrix of order n, there are n! (factorial of n) different permutation matrices.

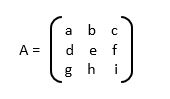
### **Determinant**

The Determinant of a square matrix is a scalar value computed from its elements.

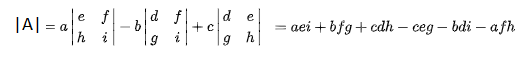
For a square matrix A of order 2:



the Determinant of A (|A|) is:  |A| = p \* s – q \* r.

For a square A matrix of order 3:

the Determinant of A (|A|) is:



### **Matrix Inversion**

Consider the matrices A and B of dimension (n, n). Let In be the identity matrix of order n.

If A \* B = B \* A = In, then B is called the inverse of A and is denoted as A-1.

AA-1 = A-1 A = In

### **Orthogonal Matrix**

Consider matrix A of dimension (n, n) where, A is called an Orthogonal matrix if,

A.AT=AT.A=In

where, In is the Identity matrix.

For an orthogonal matrix A,

A-1=AT

### **Linear Dependence**

Consider the vectors v1, v2, …., vk  and scalars c1, c2, …., ck. The vectors are **linearly dependent** if and only if,

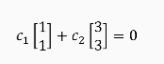


and at least one of the scalars c1, c2, … ck is non-zero.

Example: Define two vectors v1 and v2 and two scalars c1 and c2 as follows:



c1 = -3 and c2 = 1

Now,

Thus, the vectors v1 and v2 are **linearly dependent**.

Example: Define two vectors v1 and v2 as follows:

Here,



when C1=0 and C2=0.

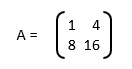
Hence, the vectors v1 and v2 are NOT linearly dependent. These are **linearly independent** vectors.

### **Rank of a Matrix**

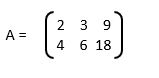
The rank of matrix A of dimension (m, n) is the number of linearly independent rows (or columns) of A.

Note: Rank of a non-zero matrix is always greater than 0 (a strictly positive integer).

Example: Consider the matrix A:



Consider the first and second row of A. They are linearly independent. So, the rank of A is 2.

Example: Consider the matrix A:

Here both the rows of A are linearly dependent. That is, the second row is 2 times the first row. So, the rank of A is 1.

If A is a matrix of dimension (m, n), then

Rank A ≤ min(m, n)

If Rank(A) = min(m, n) then A is called a **full rank matrix**.

# Importing required libraries (numpy as np)

i***mport numpy as np***

***# Creating Matrices***

***A = np.array([[1,1.0],[3,3]])***

***print('The matrix A is:\n', A)***

***B = np.eye(4)***

***print('The matrix B is:\n', B)***

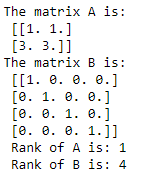
***# Finding the rank of matrix a using np.linalg.matrix\_rank() function***

***print(" Rank of A is:",np.linalg.matrix\_rank(A))***

***# Finding the rank of matrix b using np.linalg.matrix\_rank() function***

print(" Rank of B is:",np.linalg.matrix\_rank(B))

**OUTPUT**



### **Introduction - Linear Transformations**

Consider a matrix A of dimension (m, n) and a column vector x of dimension n. If you multiply the vector x with matrix A, it results into a new column vector Ax of dimension m. This operation is an example of a **transformation**.

A transformation T is said to be linear, if it satisfies the following properties:

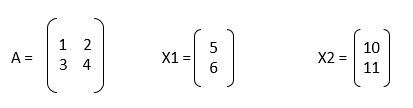
1.    T(x1 + x2) = T(x1) + T(x2)

2.    T(aX1) = aT(x1)

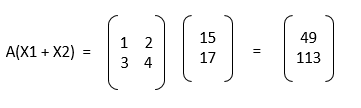
Where, x1 and x2 are vectors and a is a scalar.

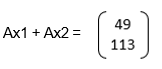
Let us understand the linear transformation with the following example.

Example: Consider a matrix A and vectors x1 and x2 as follows:



Let us now check if the given matrix and vectors satisfies the first property: A(x1 + x2) = Ax1 + Ax2.





Then, A(ax1) = aA(x1).

In the previous example, you have seen that the transformation involved multiplying a matrix with the vectors. This transformation is called **matrix transformation**. This satisfies the linearity property. Therefore, it is a linear transformation.

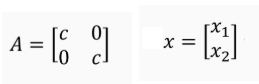
**Note:** Every matrix transformation is a linear transformation.

Next, we will discuss few more types of matrix transformations like:

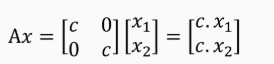
* Stretching
* Reflection
* Rotation
* Projection

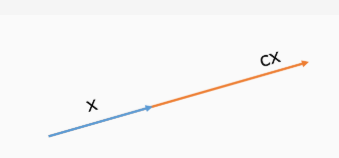
### **Stretching**

Consider a matrix A and a vector x as follows:



The matrix A stretches the vector x up to c units when A is multiplied with x.

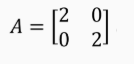


Graphically, it can be represented as:

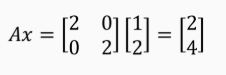
xample: Suppose x is the following vector,



If it is to be scaled by 2 units, then multiply the following matrix A with x,



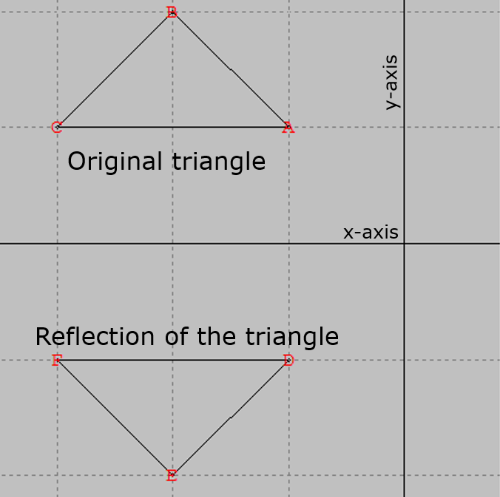
Then,

,

### **Reflection**

Reflection is a transformation that deals with reflecting an object across a reference axis.

Consider a triangle whose vertices are: A(-1, 1), B(-2, 2) and C(-3, 1). This triangle is then reflected along the x-axis to form an inverted image of itself as shown in the figure below:



### **Rotation**

Rotation of an object by θ degrees is characterized by the following matrix:



Where, θ represents the angle in a clockwise direction.

Example:

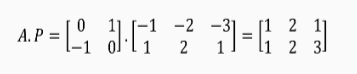
Suppose the vertices of the triangle are represented as:



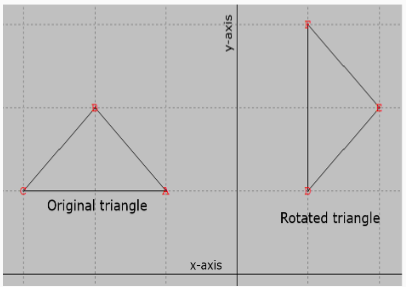
f you want to rotate this triangle by 90 degrees clockwise, putting the value of θ as 90, matrix A reduces to:



Then, the rotated triangle is:



Graphically, rotation is represented as:

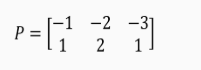


### **Projection**

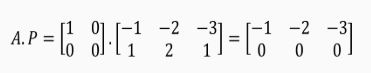
Projection of an object along the x-axis is characterized by the following matrix:



The projection of the following triangle,



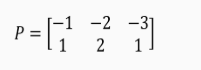
along the x-axis is:



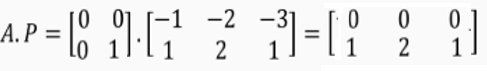
Projection along the y-axis is characterized by the following matrix:



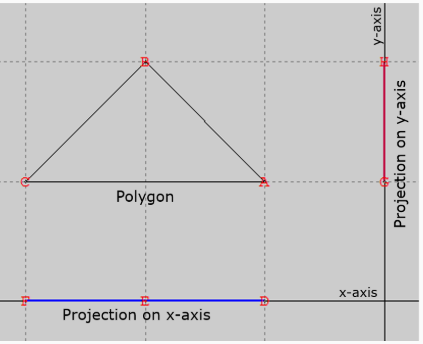
The projection of the following triangle,



along the y-axis is:

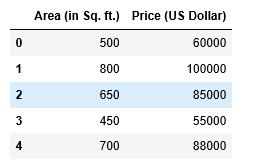


Graphically, it is represented as:

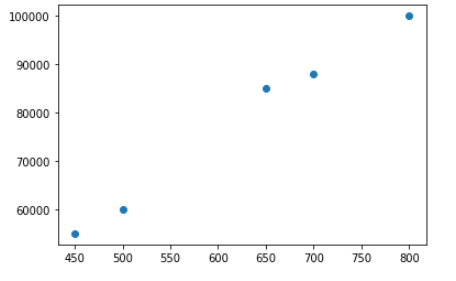


### **Introduction – Linear Equations**

Consider a scenario where you need to predict the price of a house given its area in square feet. The prediction can be made based on historical data of the houses sold in that region as given below:



Predicting the price of a house can be achieved by finding the relationship between Area and Price from the given data. The following is a scatter plot of Area and Price:



This plot indicates that there is a linear relationship between the variables Area and Price. And the linear relationship is represented by an expression as shown below:

Y = β0 + β1x1 + ϵ  ------- (1)

where,

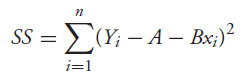
* Y is the target variable ‘Price’
* x1 is the variable ‘Area’
* β0, β1 are coefficients
* ϵ is the random error

When there are more independent variables in the dataset, the target variable can be in general represented as:

Y = β0 + β1x1 +…+ βrxr + ϵ

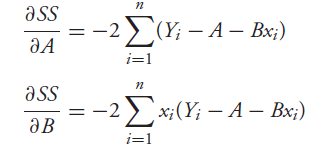
To find the optimal values of the coefficients β0 and β1, you can proceed as follows:

1. Find the sum of squares of the errors as given below:



n the above expression, A and B are estimators of the coefficients β0 and β1.

2. To find the minimum value of this error (SS), you need to differentiate the sum of square errors with the coefficients A and B and equate to 0. The differentials are:



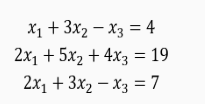
Equating the above expressions to 0, we get two linear equations in two variables A and B.

This can be generalized to r unknown coefficients and the process will result in r equations. Thus, this collection of ‘r’ linear equations is called System of Linear Equations. The values of the coefficients can be obtained by solving the System of Linear Equations.

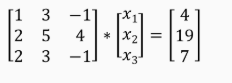
### **Solving a System of Linear Equations**

Solving a System of Linear Equations implies finding the value of the variables such that each of the equations is satisfied.

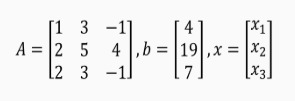
Consider a System of Linear Equations as shown below:



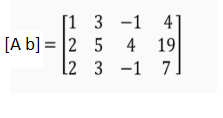
This System of Linear Equations can be represented in the form Ax = b as:



Where,



n general, the matrix A is of dimension (m, n), x is a column vector of dimension n and b is a column vector of dimension m. A is called the Coefficient matrix and the matrix [A b] is called the Augmented matrix.



Note: For a given coefficient matrix A of dimension (m, n), if m = n and A is a Full Rank matrix, then, there exists precisely one solution for the System of Linear Equations.

In the previous example of the System of Linear Equations, dimension of A is  (3, 3) and its rank is 3. Therefore, there exists precisely one solution for the Linear Equations.

A System of Linear Equations can be solved using the solve() function of linalg module of the library NumPy. The following code snippet illustrates this:

***# import the library numpy***

***import numpy as np***

***#Define the coefficient matrix***

***matrix\_1 = np.array([[1, 3, -1],***

***[2, 5, 4],***

***[2, 3, -1]])***

***# Define the vector***

***matrix\_2 = np.array([4,19,7])***

***# Find the solution for the system of equations using the solve() method***

***x= np.linalg.solve(matrix\_1, matrix\_2)***

***print("The value of x1 is: ",x[0])***

***print("The value of x2 is: ",x[1])***

***print("The value of x3 is: ",x[2])***

OUTPUT



### **Algorithms to solve a system of linear equations**

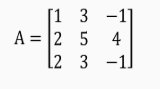
The two commonly used algorithms to solve a System of Linear Equations are:

* Gaussian Elimination
* Cramer’s Rule

### **inding Inverse**

Gauss Jordan technique can be used to find the inverse of a square matrix.

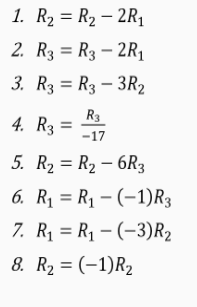
Let us find the inverse of the following matrix A of dimension (3, 3) using Gauss Jordan method:



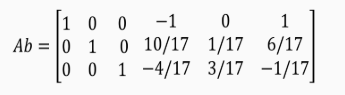
Then, augment the matrix A with Identity matrix b of dimension (3, 3) as shown below:



Perform the following operations in a sequence:



The below matrix is the result of the row operations:



n this matrix, the three columns on the extreme right represent inverse of matrix A.

There are functions in the library NumPy to find the inverse of a given square matrix. It is illustrated below:

***#import numpy library***

***import numpy as np***

***#Create matrix***

***matrix\_1 = np.array([[1, 3, -1],***

***[2, 5, 4],***

***[2, 3, -1]])***

***#Finding the Inverse of the matrix***

***print(np.linalg.inv(matrix\_1))***

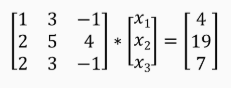
OUTPUT



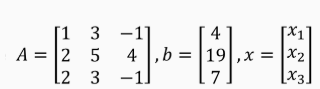
### **Cramer’s Rule**

Cramer’s Rule is a method to solve a System of Linear Equations. Unlike Gaussian Elimination, this rule helps in finding solutions to a subset of variables rather than solving the entire System of Linear Equations. This is helpful when linear equations contain large number of variables.

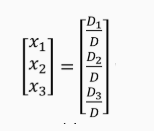
Example: Consider the following System of Linear Equations of the form Ax=b:



Let,

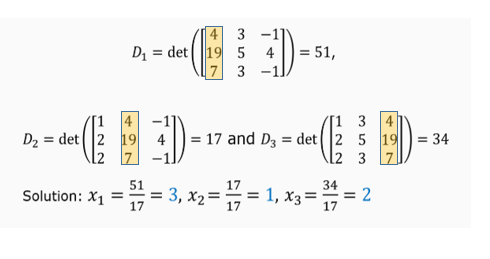


Then, the solution for the above System of Linear Equations is:



where, D is the determinant of A, and Di is the determinant of the matrix formed by replacing column i of A by b.

Here, D = det(A) = 17,



### **Solving a System of Linear Equations when the number of variables and the number of equations is not equal**

onsider a System of Linear Equations Ax=b, where A is of dimension (m, n) and m ≠ n, then the system is called

* Under - Determined System, when m < n
* Over - Determined System, when m > n

Given Ax=b, if A is invertible (inverse exists) then the solution of the System of Linear Equations is,



o find the inverse of A, when A is non-square, the generalized inverse technique needs to be used. In this case,



Where, A-G represents the generalized inverse of A (also known as the pseudo inverse).

**Note:** In NumPy, the pinv() from linalg module is used to find the generalized inverse of a matrix.